CS 433 — Project 1

Léo Paoletti, Thomas Kuntz, Gabriel Alejandro Jiménez Calles

***Abstract*—Data from US health surveys allows us to study the characteristics of patients who suffered from coronary heart disease (CHD) or myocardial infarction (MI). Our goal is to build a model based on this data to help detect patients at risk earlier. We present a pipeline from data exploration and cleaning up to cross-validation of hyper parameters allowing us to find which model among a range of considered models performs best and showing how our results to be replicated. We achieve an F1 score of and an accuracy of locally. Through AI crowd with the same model that we present in this paper, we reach an F1 score of 0.353 and an accuracy of 0.769.**

I. Introduction

The samples classified as negative are patients who did not have MICHD and the samples classified as positive are patients who did have MICHD. The models we considered for our analysis were logistic regression (regularized or not), least squares regression used for prediction, using either batch gradient descent or stochastic gradient. We evaluated the best data cleaning methods and the best ML models against each other, and performed an ablation study to compare the contributions of each intervention and to come up with an optimal pipeline.

Our best model is regularized logistic regression with $\gamma = 1.3$ and $\lambda = 5 \times 10^{-8}$.

II. Exploratory Data Analysis

A. Measuring the imbalance in the dataset

Imbalances in the representation of target classes can lead to biases in the algorithm, which minimizes the loss by over-predicting the majority class. It results in a worsening of the model's ability to classify samples of the minority class.

We measure the imbalance in our dataset to observe that 92% of our training samples are classified as negative while only 8% of our training samples are classified as positive.

Our goal is to detect patients at risk, so we need to correct this class imbalance.

B. Measuring the distribution of missing values among features

Missing values cannot be used in our learning algorithms and need to be discarded or imputed depending on the type of the feature considered. We limit ourselves to univariate missing values imputation because we are only allowed to use Numpy. Visualizing the number of features by percentage of missing values, we noticed that using a threshold of maximum 50% of missing values allowed us to keep 54% of features.

C. Defining the threshold of unique values that separate categorical and ordinal features from numerical features

Categorical and ordinal features are cleaned through different steps compared to numerical features, so we need to distinguish them. We used the distribution of unique values per feature in the training dataset and manual inspection afterwards to define a threshold. We use a threshold of 53 unique values for the features considered, which corresponds to the maximum number of unique values for a categorical or ordinal feature present in the feature _STATE.

Below 53 unique values, a feature is considered categorical or ordinal, and strictly above it is considered numerical.

III. Data Cleaning and rebalancing

Given that our dataset has 321 features, each with their own specific encoding, we defined heuristics to clean our data based on the requirements from our exploratory data analysis.

A. Handling missing values

1) Based on the distribution of the number of features by percentage of missing values, we define a threshold of percentage of missing values beyond which a feature is discarded. To reach the best trade-off between keeping as many features as possible and cleaning our data from unusable features, we defined a threshold of 50% of missing values, that we assume are represented by NaN. Beyond this percentage of missing values, a feature is discarded.
2) We impute the remaining missing values through univariate imputation methods:
   - For numerical features: We replace missing values by the median, it is more robust to outliers.
   - For categorical and ordinal features: We replace missing values by a unique value to distinguish them

Through this process, we cleaned the dataset from all missing values.

B. Rebalancing the data

As mentioned in II.A, imbalanced dataset lead to biased predictors. We tried to correct the imbalance of our dataset through:

1) Over-sampling the minority class and under-sampling the majority class: This led to a worse F1 score, hence we discarded that method for our project.
2) Because of the bad results from oversampling and undersampling, we decided to use Minibatch Gradient Descent for Logistic Regression, and, while building our minibatch, we sample the positive class with a higher probability. This method has the advantage of not having to modify the dataset.

Additionally, to solve this imbalance in the predictions we lowered the decision threshold of our model predicting to 0.3 to increase our chances to detect true positives at the expense of accuracy. This decision comes from the nature of the model we train: it is desirable to detect more patients at risk at the expense of more False Positives.

IV. Feature transformation

A. Encoding categorical and ordinal features

Categorical features had to be transformed for them to be usable by our model, as there is no meaningful order between the different US states. But our pipeline does not differentiate between categorical and ordinal features: hence we add a one-hot encoding of every feature considered categorical or ordinal and we keep the original feature to save the notion of order between the ordinal values.

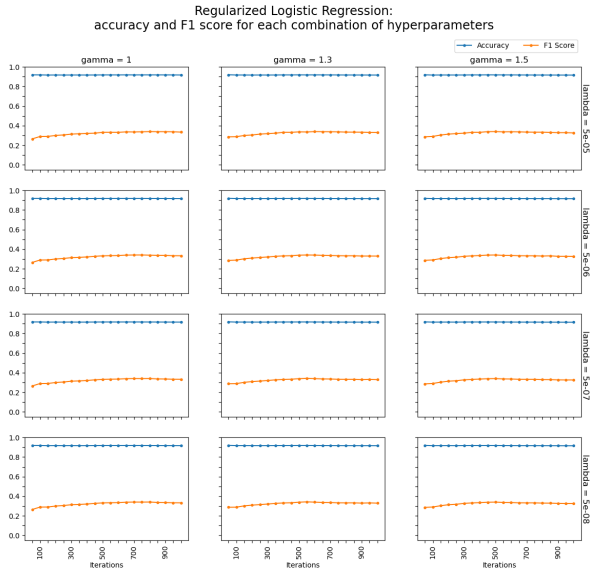

Figure 1. Results of cross-validation

B. Standardizing and handling outliers

Values that are not standardized make the learning of the model sub-optimal as the gradient oscillates. Outliers introduce bias in the predictions.

- Some categories had large values, so we standardized them, which solved the numeric errors. We standardized the values of categorical, ordinal and numerical features to allow for a better learning process.
- We used a Z-Score threshold of 3 beyond which outliers are clipped to the 10th percentile or 90th percentile of feature values.

V. Feature selection

Not all features improve prediction, because of correlations. To solve this and allow better learning, we use two feature selection methods:

- We build a correlation matrix and discard features until no entries in said matrix are higher than a certain threshold. We choose the threshold manually to allow us to keep enough features (70%), so that without increasing computation times too much.
- We do a regularized logistic regression, which forces the model to give more weight to highly predictive features, and close to 0 weight to less predictive features. We train our model using all the features a first time, then compare the absolute values of the weights and get rid of the features that have an associated weight that has a low enough absolute value. We use the threshold of absolute value that gives the best F1 score when tested on a test set.

VI. Generalization and Cross-Validation for Hyperparameter selection

We implemented a k-fold cross-validation scheme, testing the possible combinations of hyperparameters for the models we within selected ranges, and picked the ones that led to the best result on the averaged tests across the k-folds.

VII. Discussion

This approach could be further extended by automating more choices through programmatic data analysis. For example, we could include new candidate feature transformations that our pipeline could evaluate, such as polynomial feature expansion. We could include new models to test, such as SVM models or neural networks. Many more methods could be added to our pipeline without changing it fundamentally, which would enlarge the search space for the dataset preparation methods, as well as the model selection.

VIII. Ablation Study

We aim to perform an ablation study to understand the impact of each part of our pipeline on the performance of our model. We will consider the following elements of our work.

A. Data balancing by oversampling or subsampling.

This gave us high F1 scores (higher than 0.7) because of the balanced data, but we didn't perform well on the test set. Hence we corrected the imbalance through a modified decision threshold and a fix in the training process of the mini-batch gradient descent.

B. Feature selection with regularized logistic regression.

We managed to reduce the number of features of our data, which improved our training times, but our F1 scores dropped to 0.32, and our accuracy also suffered.

C. Feature selection with a correlation study.

This approach produced the best results, but our training times exploded, as we still kept a significant number of features.